JACKSON LAWRENCE
00000070612

LOUIS GABRIEL HERNANDES
00000070250

PARTICLE SWARM OPTIMIZATION

OSCAR JIRO HARLISON
00000072786

IGNATIUS STEVEN
00000070642

DEFINITION

- **Particle Swarm Optimization (PSO)** adalah sebuah algoritma metaheuristik dari cabang **Evolutionary Algorithms (EA)** untuk memecahkan masalah-masalah optimasi kompleks
- Terinspirasi dari pergerakan dan behavioral kawanan organisme, PSO awalnya memang dikembangkan untuk memimik pergerakan kawanan burung dan ikan, tetapi setelah penyempurnaan dan observasi lebih lanjut, didapatkan PSO berguna untuk melakukan optimasi

SWARM INTELLIGENCE

- Secara ilmiah, teknik dan konsep yang digunakan dalam PSO adalah **Swarm Intelligence (SI)**
- Mengimitasi kawanan organisme, SI terdiri dari suatu populasi agen-agen yang saling berinteraksi dengan sesama dan lingkungannya
- Setiap agen merupakan individu yang dapat berperilaku secara mandiri, tetapi secara bersamaan akan menyesuaikan dengan posisi kawanannya
- Beberapa algoritma lain yang menggunakan SI adalah Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Firefly Algorithm (FA), etc.

WHY USE PSO?

- **Simplisitas dan kemudahan** penerapan konsep dan teknik
- Dibandingkan banyak algoritma metaheuristik lain, beberapa observasi mendapatkan **PSO lebih superior** secara scalability, kompleksitas, akurasi, efisiensi, dan kecepatan konvergensi

APPLICATIONS

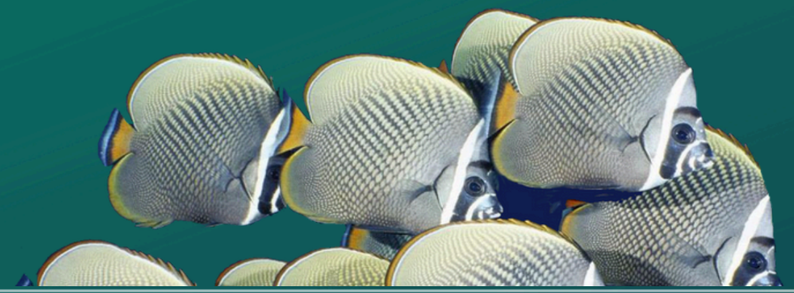
- Energy Store Optimization
- Risk Assessment
- Water Quality Monitoring
- Traveling Salesman Problem
- Scheduling
- Smart Home and Building
- Optimization

HOWEVER....

- Layaknya semua algoritma metaheuristik, jawaban **tidak deterministik**, sehingga tidak selalu sama atau **tidak menjamin jawaban ter-optimal**; untuk hal yang sama, tahap mutasi dapat seperti sebuah double-edged sword
- Risiko **konvergensi prematur** akibat minimnya randomness dan penekanan terlalu banyak di global best
- Berkemungkinan terjebak dalam minimum lokal / daerah hasil tidak dapat dijangkau oleh partikel
- Efektivitas PSO yang dapat terhambat karena **kompleksitas komputasionalnya yang tinggi**

HOW IT WORKS

- Setiap agen merepresentasikan suatu solusi untuk suatu masalah dan didefinisikan oleh posisi, kecepatan, dan nilai fitness-nya
- Setiap agen akan mengeksplorasi ruang pencarian dan akan mengetahui solusi terbaik yang pernah ditemukannya (local best) dan yang pernah ditemukan oleh seluruh kawanan (global best)
- Dari informasi local and global best, setiap agen dapat menyesuaikan posisi dan kecepatan mereka masing-masing sehingga memaksimalkan eksplorasi ruang pencarian



OVERALL CONCEPT

TERMINOLOGY & KEY COMPONENTS

Populasi / Banyak Partikel

01

Populasi adalah kumpulan dari partikel atau agen; **agen/partikel** sendiri adalah individu yang mewakili suatu kandidat solusi

Fitness

02

Nilai kesesuaian atau keoptimalan suatu solusi dengan tujuan

Bobot Inersia

03

Semakin tinggi semakin mempromosikan eksplorasi global, semakin rendah mempromosikan eksploitasi lokal/yang sudah dikenal baik

Koefisien Sosial & Kognitif

04

Koefisien kognitif semakin tinggi semakin mempromosikan pembelajaran individu, sosial semakin tinggi semakin mempromosikan pembelajaran bersama

Ukuran Kawanan

05

Memengaruhi variasi solusi dan kecepatan konvergensi algoritma; semakin besar semakin bervariasi tapi berat secara komputasi

Velocity

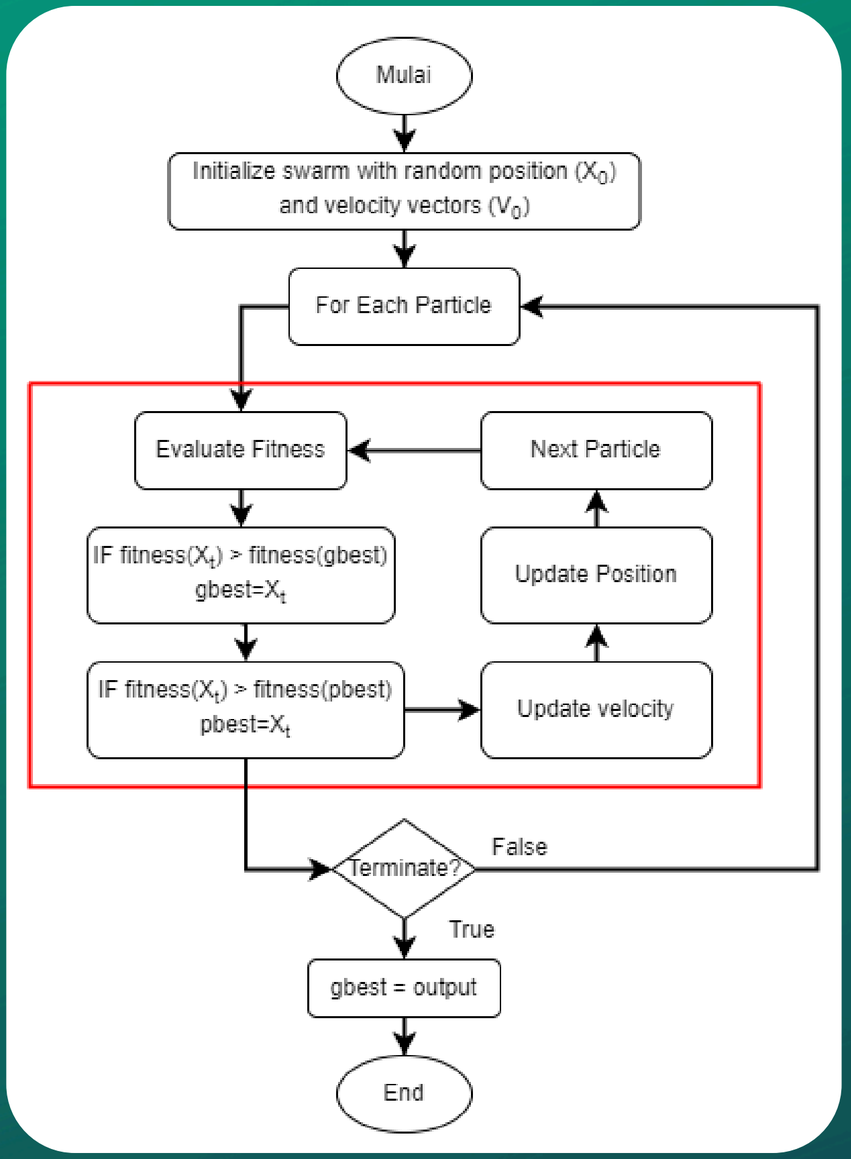
06

Tendensi perubahan sebuah agen/partikel di ruang solusi, dipengaruhi oleh koefisien sosial dan kognitif

Personal & Global Best

07

Personal best merujuk kepada solusi dengan fitness terbaik yang pernah didapatkan sebuah agen/partikel, global best merujuk kepada solusi terbaik yang pernah ditemukan

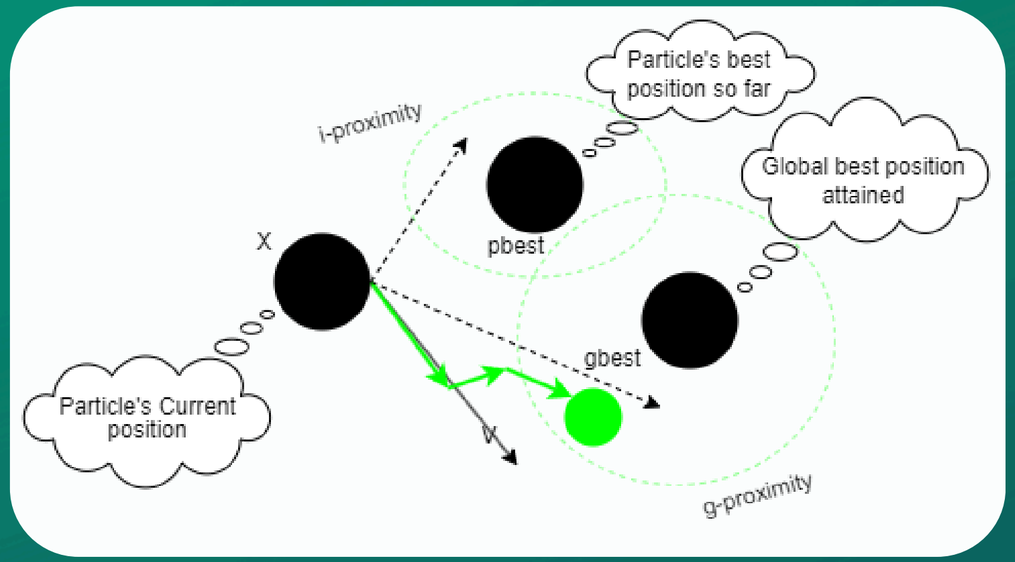


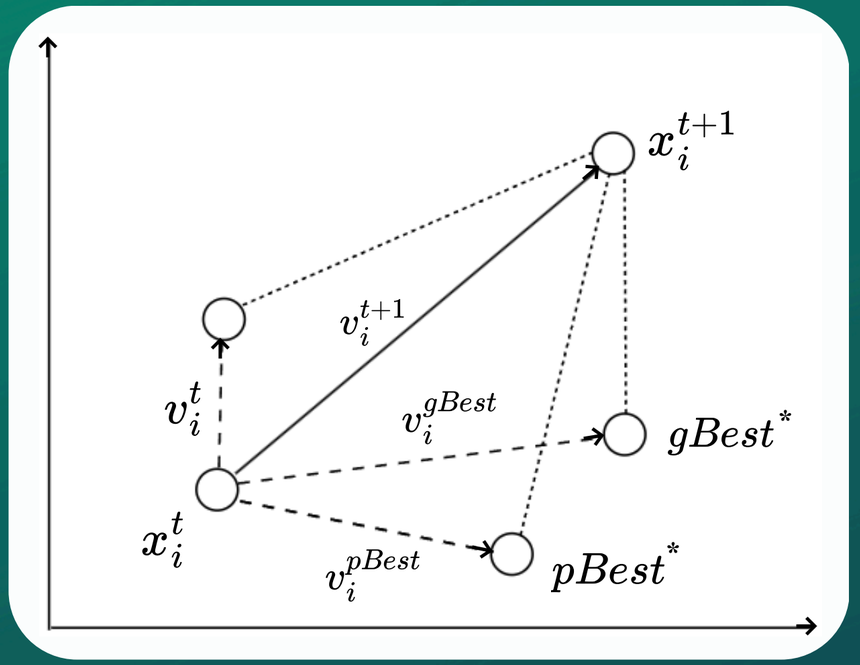
PSEUDOCODE

Particle Swarm Optimization Algorithm

```
initialize_swarm()

while not termination_criteria():
  for particle in swarm:
    fitness = evaluate_fitness(particle.position)

    if fitness > gbest.fitness:
      gbest.position = particle.position
      gbest.fitness = fitness

    if fitness > particle.pbest.fitness:
      particle.pbest.position = particle.position
      particle.pbest.fitness = fitness

    particle.velocity = update_velocity(particle, gbest)
    particle.position = update_position(particle)

  if termination_criteria():
    break
output = gbest.position
```

SEARCH MECHANISM





TIME COMPLEXITY

$$O(N * D * T)$$

, dimana :

- N = Jumlah Partikel

SPACE COMPLEXITY

$$O(N * (D + 2))$$

- D = Dimensi (ukuran) tempat pencarian
- T = Jumlah maksimal iterasi

UPDATE VELOCITY

$$v_i\left[t+1\right] = w \times v_i\left[t\right] + c_1 r_1\left(pBest_i\left[t\right] - x_i\left[t\right]\right) + c_2 r_2\left(gBest\left[t\right] - x_i\left[t\right]\right)$$

Informations:

- **Larger w = Greater Global Search Ability**
- **Smaller w = Greater Local Search Ability**
- **c1 > c2 = Greater Global Search Ability**
- **c1 < c2 = Greater Local Search Ability**

UPDATE POSITION

$$x_i\left[t+1\right] = x_i\left[t\right] + v_i\left[t+1\right]$$

, where:

- i = Partikel ke-i
- t = Iterasi ke-t
- x = Posisi partikel & v = Kecepatan partikel
- c_1 = Koefisien pembelajaran kognitif, yang mengatur seberapa jauh partikel tertarik pada posisi terbaiknya sendiri
- c_2 = Koefisien pembelajaran sosial, mengatur seberapa jauh partikel tertarik pada posisi terbaik global (Global best)
- r_1 = r_2 = rand(0, 1) | r_1 untuk pencarian pribadi dan r_2 untuk pencarian global dengan nilai antara 0 dan 1

Notes: w, c_1, dan c_2 biasanya diketahui dalam soal permasalahan

INERTIA WEIGHT

$$w = w_{max} - \left(\frac{w_{max} - w_{min}}{t_{max}}\right) \times t$$

- $pBest_i$ = Posisi terbaik yang pernah dicapai oleh partikel ke-i (Best Personal Position / Best Local Position)
- gBest = Posisi terbaik yang ditemukan oleh seluruh partikel dalam populasi (Best Global Position)
- w = inertia weight (Biasanya 0.4 ≤ w ≤ 0.9), mengontrol pengaruh dari kecepatan sebelumnya dalam menjaga keseimbangan antara **eksplorasi** (mencari area baru dalam ruang pencarian) dan **eksploitasi** (memanfaatkan area yang sudah ditemukan untuk mendapatkan solusi terbaik).

OPTIMIZING A FUNCTION

$$f(x) = -x^2 + 2x + 11 \quad -2 \leqslant x \leqslant 2$$

I. Initialize x & v untuk tiap partikel i = 1,2,3,4 dan hitung nilai f(x)

Initial Positions:

x1 = -1.5 | x2 = 0 | x3 = 0.5 | x4 = 1.25

Initial Velocities:

v1 = v2 = v3 = v4 = 0

Weight Inertia:

w = 0.8

Acceleration Coefficients:

c1 = c2 = 2.05

Nilai f(x) tiap partikel:

$f(x1) = -(-1.5)^2 + 2(-1.5) + 11 = 5.75$
$f(x2) = -(0)^2 + 2(0) + 11 = 11$
$f(x3) = -(0.5)^2 + 2(0.5) + 11 = 11.75$
$f(x4) = -(1.25)^2 + 2(1.25) + 11 \approx 11.94$

II. Update Personal / Global Best Position, Velocity, & Position untuk partikel x1

Nilai gBest:

gBest = 11.94 dengan x4 = 1.25, karena nilai f(x4) tertinggi

Update Velocity:

$v_i[t+1] = w * v_i[t] + c_1r_1(pBest[t] - x_i[t]) + c_2r_2(gBest[t] - x_i[t])$
$v_1[1] = w * v_1[0] + c_1r_1(pBest[0] - x_1[0]) + c_2r_2(gBest[0] - x_1[0])$
$v_1[1] = 0.8*0 + 2.05*0.3*(-1.5-(-1.5)) + 2.05*0.6*(1.25-(-1.5))$
$v_1[1] = 3.3825$

Update Position:

$x_i[t+1] = x_i[t] + v_i[t]$
$x_1[1] = x_1[0] + v_1[1]$
$x_1[1] = -1.5 + 3.3825$
$x_1[1] = 1.8825 \approx 1.88$

Nilai pBest:

pBest(x1) = -1.5
pBest(x2) = 0
pBest(x3) = 0.5
pBest(x4) = 1.25

III. Update Velocity & Position untuk semua partikel

Partikel x2:

$v_2[1] = w*v_2[0]+c_1r_1(pBest[0]-x_2[0])+c_2r_2(gBest[0]-x_2[0])$
$v_2[1] = 0.8*0+2.05*0.2*(0-0)+2.05*0.6*(1.25-0)$
$v_2[1] = 1.5375$

$x_2[1] = x_2[0] + v_2[1]$
$x_2[1] = 0 + 1.5375$
$x_2[1] = 1.5375 \approx 1.54$

Partikel x3:

$v_3[1] = w*v_3[0]+c_1r_1(pBest[0]-x_3[0])+c_2r_2(gBest[0]-x_3[0])$
$v_3[1] = 0.8*0+2.05*0.4*(0.5-0.5)+2.05*0.1*(1.25-0.5)$
$v_3[1] = 0.15375$

$x_3[1] = x_3[0] + v_3[1]$
$x_3[1] = 0.5 + 0.15375$
$x_3[1] = 0.65375 \approx 0.65$

OPTIMIZING A FUNCTION

$$f(x) = -x^2 + 2x + 11 \quad -2 \leqslant x \leqslant 2$$

$x_1[2]$ & $x_2[2]$ = 2 karena sudah berada di luar jangkauan $-2 \leq x \leq 2$

IV. Lanjutan bagian III. & Update New Solution
- f(x) dari posisi partikel untuk iterasi 1

Partikel x4:

$v_4[1] = w*v_4[0]+c_1r_1(pBest[0]-x_4[0])+c_2r_2(gBest[0]-x_4[0])$
$v_4[1] = 0.8*0+2.05*0.9*(1.25-1.25)+2.05*0.2*(1.25-1.25)$
$v_4[1] = 0$

$x_4[1] = x_4[0] + v_4[1]$
$x_4[1] = 1.25 - 0$
$x_4[1] = 1.25$

Hasil nilai f(x) baru untuk iterasi 1:

$x_1[1] = 1.88 \rightarrow f(x1) = -(1.88)^2+2*1.88+11 \approx 11.23$
$x_2[1] = 1.54 \rightarrow f(x2) = -(1.54)^2+2*1.54+11 \approx 11.69$
$x_3[1] = 0.65 \rightarrow f(x3) = -(0.65)^2+2*0.65+11 \approx 11.88$
$x_4[1] = 1.25 \rightarrow f(x4) = -(1.25)^2+2*1.25+11 \approx 11.94$

↳ Semua berada pada jangkauan $-2 \leq x \leq 2$, sehingga digunakan nilainya

V. Update Personal & Global Best Solution
- dari hasil iterasi 0 dan masuk iterasi 1

Check fitness Old_gBest & New_gBest:

Check (5.75) < (11.23)
(11) < (11.69)
(11.75) < (11.88)
(11.94) = (11.94)

∴ gBest = 11.94, dengan x4 = 1.25

Ulangi bagian 2, 3, 4 untuk iterasi 1

$v_1[2] = 3.4809$ $v_4[2] = 0$
$x_1[2] = 5.36$ $x_4[2] = 1.25$

$v_2[2] = 0.8856$
$x_2[2] = 2.43$

$v_3[2] = 0.738$
$x_3[2] = 1.33$

Update pBest:

pBest(x1) = 1.88
pBest(x2) = 1.54
pBest(x3) = 0.65
pBest(x4) = 1.25

VI. Update New Solution f(x) dari posisi
- partikel untuk iterasi 2

Hasil nilai f(x) baru untuk iterasi 2:

$x_1[2] = 2 \rightarrow f(x1) = -(2)^2+2*2+11 = 11$
$x_2[2] = 2 \rightarrow f(x2) = -(2)^2+2*2+11 = 11$
$x_3[2] = 1.33 \rightarrow f(x3) = -(1.33)^2+2*1.33+11 = 11.89$
$x_4[2] = 1.25 \rightarrow f(x4) = -(1.25)^2+2*1.25+11 = 11.94$

Check fitness Old gBest & New gBest:

Check (11.23) > (11)
(11.69) > (11)
(11.88) < (11.89)
(11.94) = (11.94)

∴ gBest = 11.94, dengan x4 = 1.25

↳ Dilakukan sebanyak n iterasi dan apabila tiap iterasi gBest tidak menghasilkan perubahan signifikan, maka nilai optimalnya adalah gBest itu sendiri dengan pBestnya sendiri karena dianggap telah mencapai keadaan konvergen / solusi optimal.

TRAVELING SALESMAN PROBLEM

		A	B	C	D	E
		Kota 1	Kota 2	Kota 3	Kota 4	Kota 5
A	Kota 1	0	132	217	164	58
B	Kota 2	132	0	290	201	79
C	Kota 3	217	290	0	113	303
D	Kota 4	164	201	113	0	196
E	Kota 5	58	79	303	196	0

Misalkan ada 3 partikel mewakili tiap rute.

Misalkan nilai:

- $w = 0.9$
- $v = 0.1$
- $c_1r_1 = 0.4$
- $c_2r_2 = 0.6$

Inisialisasi posisi / pBest awal (Rute awal secara acak):

- Partikel 1: [0.8023, 0.0732, 0.0392, 0.5178, 0.9350]
- Partikel 2: [0.4242, 0.5910, 0.9463, 0.9942, 0.4795]
- Partikel 3: [0.7289, 0.9102, 0.7637, 0.8549, 0.2318]

Urutkan posisi / Rute awal secara ascending:

- Partikel 1: [0.0392, 0.0732, 0.5178, 0.8023, 0.9350]
- Partikel 2: [0.4242, 0.4795, 0.5910, 0.9463, 0.9942]
- Partikel 3: [0.2318, 0.7289, 0.7637, 0.8549, 0.9102]

Ubah perubahan posisi bilangan acak:

Partikel 1: [0.8023, 0.0732, 0.0392, 0.5178, 0.9350]
A B C D E
[0.0392, 0.0732, 0.5178, 0.8023, 0.9350] **ΣJarak: 1016**
C B D A E

Partikel 2: [0.4242, 0.5910, 0.9463, 0.9942, 0.4795]
A B C D E
[0.4242, 0.4795, 0.5910, 0.9463, 0.9942] **ΣJarak: 704**
A E B C D

Partikel 3: [0.7289, 0.9102, 0.7637, 0.8549, 0.2318]
A B C D E
[0.2318, 0.7289, 0.7637, 0.8549, 0.9102] **ΣJarak: 668 → gBest**
E A C D B

Update kecepatan:

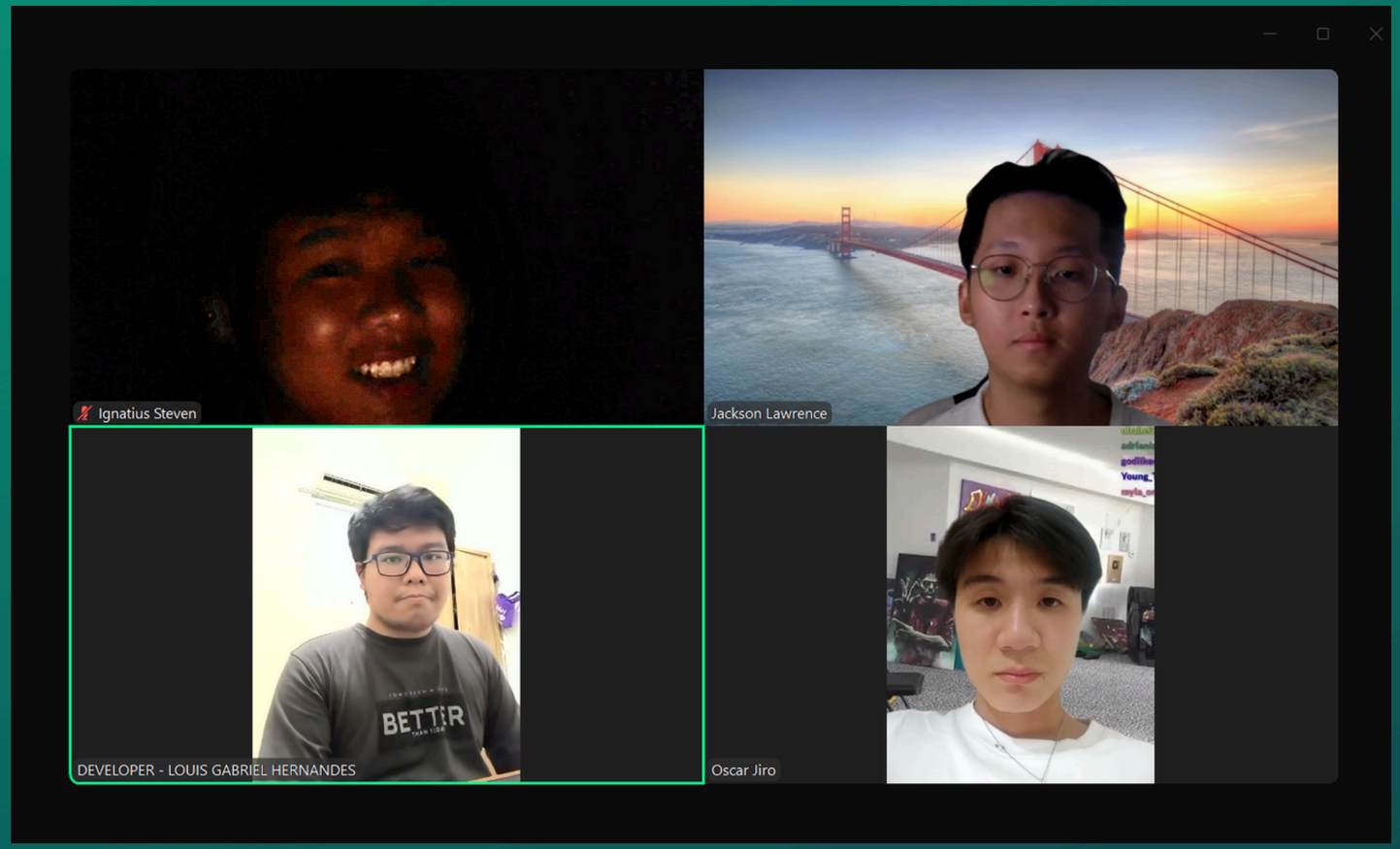
$v_i[t+1] = w * v_i[t] + c_1r_1(pBest[t] - x_i[t]) + c_2r_2(gBest[t] - x_i[t])$
$v_1[1] = 0.9*([0.1, 0.1, 0.1, 0.1, 0.1]) + 0.4*([0.8023, 0.0732, 0.0392, 0.5178, 0.9350]-[0.8023, 0.0732, 0.0392, 0.5178, 0.9350])+0.6([0.7289, 0.9102, 0.7637, 0.8549, 0.2318]-[0.2318, 0.7289, 0.7637, 0.8549, 0.9102]) = \alpha$

Update posisi:

$x_i[t+1] = x_i[t] + v_i[t]$
$x_1[1] = x_1[0] + v_1[1]$
$x_1[1] = [0.8023, 0.0732, 0.0392, 0.5178, 0.9350] + \alpha = \beta$

Notes:

Perhitungan dilanjuti seperti metode ini dengan iterasi maksimal apabila nilai gBest sudah tidak berubah secara signifikan.



Links:

- https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/
- https://lp2m.uma.ac.id/2023/04/01/particle-swarm-optimization-pso-definisi-dan-penjelasannya/
- https://en.wikipedia.org/wiki/Particle_swarm_optimization
- https://thisisrishi.medium.com/understanding-particle-swarm-optimization-pso-from-basics-to-brilliance-d0373ad059b6
- https://chatgpt.com/share/66eae344-b5dc-8008-9f30-693720a134de

Paper:

- https://www.iitg.ac.in/rkbc/CE602/CE602/Particle%20Swarm%20Algorithms.pdf
- https://link.springer.com/article/10.1007/s11831-021-09694-4
- https://dl.acm.org/doi/abs/10.1145/2906150

Video:

- https://youtu.be/JhgDMAm-imI?si=OlCTdxdgqnplhHLq
- https://youtu.be/1g66clXrnmA?si=MCPCyAhfInBQK5M5
- https://www.youtube.com/watch?v=uwXFnzWaCY0
- https://www.youtube.com/watch?v=HmDjfL3R39M&list=PLVLAu9B7VtkaYCFbAv59r761wqx-hSx_V
- https://youtu.be/vBckxndH8tw?si=llpf8N4fptEAu02f
- https://youtu.be/vBckxndH8tw?si=1YBxNBEKxu8-HHYz